CSC 3520
Machine Learning
Florida Southern College

HW1: Probability and Bayes Classifiers
Due: Thursday, September 22, 2022

In this assignment, you will have the opportunity to:
- apply basic probability rules to simple problems
- make inferences using Bayes rule
- train a Naïve Bayes model on a real-world dataset
- practice programming in Python, including the use of numpy, scikit-learn, and matplotlib modules

1. Consider the Bayesian network pictured below. The structure of the network dictates that *B* and *C* are conditionally independent given *A*. In other words,

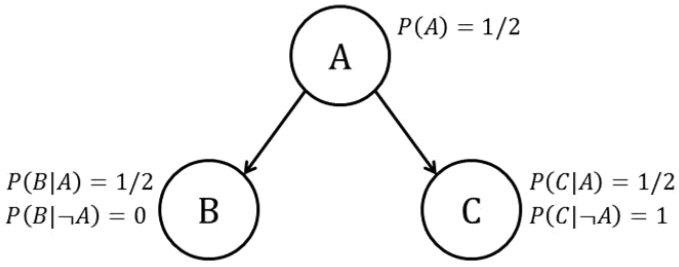
$$P(B \mid C, A) = P(B \mid A) \quad \text{and} \quad P(B \mid C, \neg A) = P(B \mid \neg A)$$
$$P(C \mid B, A) = P(C \mid A) \quad \text{and} \quad P(C \mid B, \neg A) = P(C \mid \neg A)$$

All other probability rules apply as usual.

(a) Compute $P(A, B)$.

(b) Compute $P(C)$.

(c) Compute $P(B \mid C)$.



2. Suppose you witness a nighttime hit-and-run accident involving a taxi. You know that all taxis in your city are either yellow or green. You swear, under oath, that the taxi *appeared green*. Extensive testing shows that under dim lighting conditions, discriminating between yellow and green is prone to error. Specifically, 1 out of every 3 green taxis appear yellow and 1 out of every 4 yellow taxis appear green. Given that 80% of taxis are yellow, what is the probability that the taxi was *actually green*?

   (HINT: Define A = the taxi appears green and B = the taxi is green)

3. For this exercise, you will be developing Python code to automatically classify written documents based on the words appearing in them using a Naïve Bayes model.

   Suppose a document D contains m words, labeled in the order they appear $\{X_1, X_2, \ldots, X_m\}$. The value of the random variable X_i is the word found in the ith position in the document. For instance, X_1 corresponds to the first word in the document, and its value is the word itself. Our goal is to predict the label Y for each document, which can be one of n categories. To accomplish this, we could use a Naïve Bayes model as follows:

$$P(Y|X_1 \ldots X_m) \propto P(X_1 \ldots X_m|Y)P(Y) = P(Y)\prod_i P(X_i|Y)$$

That is, each X_i is sampled from some distribution that depends on its position X_i and the document category Y. As usual with discrete data, we assume that $P(X_i|Y)$ is a multinomial distribution over some vocabulary V; each X_i can take one of many possible values corresponding to the words in the vocabulary. Therefore, in this model, we are assuming (roughly) that for any pair of document positions i and j, $P(X_i|Y)$ and $P(X_j|Y)$ may be completely different.

To illustrate these concepts more clearly, consider the following example. The words *Hello*, *Hi*, and *Dear* likely have high probabilities associated with the first position (X_1) for documents in the class *Personal Emails*, but very low probabilities as the first word (or any word, for that matter) in the class *Scientific Articles*. Furthermore, even in the *Personal Emails* class, those same words likely have much lower probabilities associated with other positions in the document (*i.e.* the word *Dear* is typically not found anywhere other than the first word).

(a) Explain in at least 2-3 sentences why it would be difficult to accurately estimate the parameters of this model with a reasonable set of documents (*e.g.* 1,000 documents, each 1,000 words long, where each word comes from a 50,000 word vocabulary).

To improve the model, we will make the additional assumption that $P(X_i|Y) = P(X_j|Y)\ \forall i, j$. In other words, we no longer discriminate between the positions of the words. We are simply concerned with the frequency of each word appearing in the document. For instance, we expect words such as *cookie*, *pasta*, and *appetizer* to appear more frequently in the document class *Restaurant Menus* than in the class *Newspapers*.

(b) Write a Python program to implement the Naïve Bayes classifier as described above using the data provided in the assignment. You should estimate $P(Y)$ using maximum likelihood estimation (MLE) and $P(X|Y)$ using maximum a posteriori (MAP) estimation. For the latter, use a Dirichlet prior distribution, $Diri(1+\alpha, \ldots, 1+\alpha)$, where $\alpha = 1/|V|$ and V is the vocabulary.

In the previous part, the Dirichlet distribution parameter α was given. In practice, if the domain knowledge is not sufficient to set prior parameters, then to avoid overfitting, those parameters should be selected based on the performance of different values on some validation set.

(c) Retrain your Naïve Bayes classifier for at least 20 different values of α between 0.00001 and 1 and report the accuracy over the test set for each value. Create a plot with α on the *x*-axis and accuracy on the *y*-axis. Use a logarithmic scale for the *x*-axis. Explain in 2-3 sentences why accuracy drops for both small and large values of α.

The data for this problem has been provided for you. There are 11,269 documents for training and 7,505 documents for testing, each belonging to one of 20 classes. Each document contains a number of words, which come from a vocabulary of 61,188 unique words. The goal in this problem is to predict the document class, given the words in a document. The relevant data files are listed below.

`vocabulary.txt` List of words in the vocabulary. The line number indicates the *wordID* in other files; for instance, the first word (*archive*) has *wordID* 1, the second word (*name*) has *wordID* 2, etc.

`newsgroups.txt` List of newsgroups (classes) from which a document may have come. The line number corresponds to the class identifier, which is used in the labels files. Class 1 is *alt.atheism*, class 2 is *comp.graphics*, and so on.

`traininglabels.txt` `testinglabels.txt`	Each line corresponds to a document from the training/testing set and contains a number referencing the label (class) for that document.
`trainingdata.txt` `testingdata.txt`	List of word counts for each document in the training/testing set. Each line is of the form "*docID wordID count*", which specifies the number of times (*count*) that a given word (*wordID*) appears in a given document (*docID*).

Starter code has also been provided for you (`naivebayes.py`). Sections that require editing are either marked by the text `***MODIFY CODE HERE***` or a variable set to `-1`. You only need to submit the code for part (b), i.e. do not include the alpha variation in your code. The output of a successful program is shown on the following page. The code you develop for part (c) will not be submitted; rather, the plot of accuracy versus alpha will indicate whether you have correctly solved that part of the problem.

Here are some functions from the imported libraries that *may* be useful:

`numpy`	`add, argmax, bincount, loadtxt, log, matmul, mean, sum, zeros`
`sklearn.metrics`	`confusion_matrix`
`matplotlib.pyplot`	`figure, semilogx, show, xlabel, ylabel`

```
$ python naivebayes.py
Document Classification using Naive Bayes Classifiers
======================
PRE-PROCESSING
======================
Loading training data...
Loading training labels...
Loading testing data...
Loading testing labels...
Loading newsgroups...
Loading vocabulary...

======================
TRAINING
======================
Estimating prior probabilities via MLE...
Estimating class conditional probabilities via MAP...

======================
TESTING
======================
Applying natural log to prevent underflow...
Counting words in each document...
Computing posterior probabilities...
Assigning predictions via argmax...

======================
PERFORMANCE METRICS
======================
Accuracy:
Confusion matrix:
```

```
(csc3520)
meicholtz@CSC005 MINGW64 ~/Documents/csc3520/assignments/hw1
$
```